PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 12/00

A2

(11) International Publication Number: WO 99/23784

(43) International Publication Date: 14 May 1999 (14.05.99)

(21) International Application Number: PCT/US98/22656

(22) International Filing Date: 26 October 1998 (26.10.98)

(30) Priority Data:
08/962,485 31 October 1997 (31.10.97) US

(71) Applicant: ORACLE CORPORATION [US/US]; 500 Oracle Parkway, Redwood Shores, CA 94065 (US).

(72) Inventors: CHOU, Tsung-Jen; 15 La Loma Drive, Menlo Park, CA 94025 (US). ADUNUTHULA, Seshu; 34543 Felix Terrace, Fremont, CA 94555 (US). ANAND, Mala; 501 Forest Avenue, Palo Alto, CA 94301 (US). SHARMA, Ankur; 951-2 Old County Road #365, Belmont, CA 94002 (US). CHIEN, Elaine; 780 Bradford Street #15, Redwood City, CA 94063 (US). NAKHODA, Shehzaad; 455 Grant Avenue #14, Palo Alto, CA 94306 (US).

(74) Agents: CARLSON, Stephen, C. et al.; McDermott, Will & Emery, Suite 300, 99 Canal Center Plaza, Alexandria, VA 22314 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: DISTRIBUTED WEB APPLICATION SERVER

(57) Abstract

A system, method, and computer readable-medium for performing operations associated with browser requests are provided. The system includes a plurality of dispatchers coupled to a plurality of web listeners. Each of the dispatchers receives from a corresponding web listener browser requests received by the corresponding web listener. The system further includes a virtual path manager and a resource manager. The virtual path manager is coupled to the dispatchers through an inter-machine communication mechanism. The virtual path manager indicates to the dispatchers which of a cartridge is associated with the browser requests. The resource manager is coupled to the dispatchers through the inter-machine communication mechanism. The resource manager is configured to assign to each dispatcher of the dispatchers an instance of a cartridge of the cartridges in response to receiving a request for an instance from the dispatcher. The dispatchers are configured to send messages through the inter-machine communication mechanism to the instances that are assigned by the resource manager to the dispatchers. The messages cause the instances to perform the operations associated with the browser requests.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

DISTRIBUTED WEB APPLICATION SERVER

FIELD OF THE INVENTION

This invention relates to server architectures in networked computer systems, more specifically to a distributed architecture for enabling the dynamic servicing to user requests across different machines.

BACKGROUND OF THE INVENTION

The World Wide Web includes a network of servers on the Internet, each of which is associated with one or more HTML (Hypertext Markup Language) pages. The HTML pages associated with a server provide information and hypertext links to other documents on that and (usually) other server. Servers communicate with clients by using the Hypertext Transfer Protocol (HTTP). The servers listen for requests from clients for their HTML pages, and are therefore often referred to as "listeners".

Users of the World Wide Web use a client program, referred to as a browser, to request, decode and display information from listeners. When the user of a browser selects a link on an HTML page, the browser that is displaying the page sends a request over the Internet to the listener associated with the Universal Resource Locator (URL) specified in the link. In response to the request, the listener transmits the requested information to the browser that issued the request. The browser receives the information, presents the received information to the user, and awaits the next user request.

Traditionally, the information stored on listeners is in the form of static HTML pages. Static HTML pages are created and stored at the listener prior to a request from a web browser. In response to a request, a static HTML page is merely read from storage and transmitted to the requesting browser. Currently, there is a trend to develop listeners that respond to browser requests by performing dynamic operations. For example, a listener may respond to a request by issuing a query to a database, dynamically constructing a web page containing the results of the query, and transmitting the dynamically constructed HTML page to the requesting browser. To perform dynamic operations, the functionality of the listener must be enhanced or augmented. Various approaches have been developed for extending listeners to support dynamic operations.

One approach to providing dynamic operations in response to requests from web browsers uses the common gateway interface (CGI). CGI is a specification for transferring information between a listener and a CGI program. A CGI program is any program designed to accept and return data that conforms to the CGI specification. The program could be written in any programming language, including C, Perl, or Visual Basic.

The CGI approach suffers from the disadvantage that a separate process (a separate instance of the CGI program) is initiated each time the specified request is received by the server. Further, CGI programs execute on the same machine as the listener that received the browser request. Consequently, receipt of a thousand such requests from different users will cause a thousand processes to be initiated on the machine running the listener, exhausting available resources on the server.

A second disadvantage of the CGI approach is that a separate process is initiated, executed and terminated for each request. Thus, if a first set of ten requests are followed by a second set of ten requests, a first set of ten processes will be initiated and terminated for the first set of requests and a second set of ten processes will be initiated and terminated for the second set of requests. CGI does not allow using the same ten processes that are used for the first ten requests to process the second ten requests to avoid overhead associated with initiating processes.

An alternative approach to providing dynamic responses to requests involves using "plug-in" extensions. A plug-in extension intercepts messages sent to the server at various stages to perform application-specific processing for a specific user request. A server-side plug-in executes in the same address space as the listener and all other server-side plug-ins. Hence, an application developer designing a plug-in must be familiar with the lower level operational details of the listener. Moreover, execution of the plug-ins in the same address space as the listener exposes the listener to security and stability risks, where a faulty plug-in may cause other plug-ins or the listener itself to crash, or perform in an unpredictable manner.

A second problem with the plug-in approach is that, similar to the CGI approach, all plug-in operations are performed on the same machine that is executing the listener. Because the tasks performed by the plug-in extensions cannot be off-loaded to other machines, the scalability of the plug-in approach is significantly limited.

SUMMARY OF THE INVENTION

A method and system for handling browser requests with a distributed web application server is provided. The distributed environment allows processes ("cartridge instances") that perform the operations specified in the browser requests to execute on different machines than the listeners that receive the requests and the browsers that issue the requests. Because the cartridge instances are on different machines than the listeners, the listeners are better insulated against faulty cartridge instances, thus enhancing the reliability and security of the system. In addition, the scalability of the system is greatly increased by spreading the processing burden of executing the cartridge instances among many machines, rather than the same machine that is executing the listener. The ability to distribute cartridge instance execution across multiple machines allows numerous types of load balancing techniques to be used in deciding when and where to spawn new cartridge instances.

According to one aspect of the invention, an operation is executed using a dispatcher that is executing on a first machine and a resource manager that is executing on a second machine. A first message is sent from the dispatcher to the resource manager. The first message identifies a particular cartridge that is able to perform the operation. A second message is sent from the resource manager to the dispatcher. The second message identifies a particular instance of the particular cartridge. The particular instance is executing on a third machine.

A third message from the dispatcher is sent to the particular instance to cause the particular instance to execute the operation. At least two of the first machine, the second machine and the third machine are separate machines.

According to another aspect of the invention, a system for performing operations associated with browser requests is provided. The system includes a plurality of dispatchers coupled to a plurality of web listeners. Each dispatcher of the plurality of dispatchers receives from a corresponding web listener of the plurality of web listeners browser requests received by the corresponding web listener.

The system further includes a virtual path manager and a resource manager. The virtual path manager is coupled to the plurality of dispatchers through an inter-machine communication mechanism. The virtual path manager indicates to the dispatchers which of a plurality of cartridges is associated with the browser requests. The resource manager is coupled to the plurality of dispatchers through the inter-machine communication mechanism.

The resource manager is configured to assign to each dispatcher of the plurality of dispatchers an instance of a cartridge of the plurality of cartridges in response to receiving a request for an instance from the dispatcher.

The plurality of dispatchers are configured to send messages through the inter-machine communication mechanism to the instances that are assigned by the resource manager to the dispatchers. The messages cause the instances to perform the operations associated with the browser requests.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1 is a block diagram of a computer system upon which an embodiment of the invention may be implemented;

Figure 2 is a block diagram of a distributed application server according to an embodiment of the invention;

Figure 3A is a portion of a flow chart illustrating steps for handling a browser request according to an embodiment of the invention;

Figure 3B is another portion of the flow chart illustrating steps for handling a browser request according to an embodiment of the invention;

Figure 4 is a block diagram of a table containing information maintained by a dispatcher according to an embodiment of the invention; and

Figure 5 is a block diagram of a table containing information maintained by a resource manager according to an embodiment of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for performing operations over a network is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

HARDWARE OVERVIEW

Figure 1 is a block diagram that illustrates a computer system 100 upon which an embodiment of the invention may be implemented. Computer system 100 includes a bus 102 or other communication mechanism for communicating information, and a processor 104 coupled with bus 102 for processing information. Computer system 100 also includes a main memory 106, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 102 for storing information and instructions to be executed by processor 104. Main memory 106 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 104. Computer system 100 further includes a read only memory (ROM) 108 or other static storage device coupled to bus 102 for storing static information and instructions for processor 104. A storage device 110, such as a magnetic disk or optical disk, is provided and coupled to bus 102 for storing information and instructions.

Computer system 100 may be coupled via bus 102 to a display 112, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 114, including alphanumeric and other keys, is coupled to bus 102 for communicating information and command selections to processor 104. Another type of user input device is cursor control 116, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 104 and for controlling cursor movement on display 112. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

The invention is related to the use of computer system 100 to perform specific operations in response to messages from browsers. According to one embodiment of the invention, the operations are performed by computer system 100 in response to processor 104 executing one or more sequences of one or more instructions contained in main memory 106. Such instructions may be read into main memory 106 from another computer-readable medium, such as storage device 110. Execution of the sequences of instructions contained in main memory 106 causes processor 104 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 104 for execution. Such a medium may take

many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 110. Volatile media includes dynamic memory, such as main memory 106. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 102. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 104 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 100 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector coupled to bus 102 can receive the data carried in the infra-red signal and place the data on bus 102. Bus 102 carries the data to main memory 106, from which processor 104 retrieves and executes the instructions. The instructions received by main memory 106 may optionally be stored on storage device 110 either before or after execution by processor 104.

Computer system 100 also includes a communication interface 118 coupled to bus 102. Communication interface 118 provides a two-way data communication coupling to a network link 120 that is connected to a local network 122. For example, communication interface 118 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 118 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 118 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 120 typically provides data communication through one or more networks to other data devices. For example, network link 120 may provide a connection

through local network 122 to a host computer 124 or to data equipment operated by an Internet Service Provider (ISP) 126. ISP 126 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 128. Local network 122 and Internet 128 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 120 and through communication interface 118, which carry the digital data to and from computer system 100, are exemplary forms of carrier waves transporting the information.

Computer system 100 can send messages and receive data, including program code, through the network(s), network link 120 and communication interface 118. In the Internet example, a server 130 might transmit a requested code for an application program through Internet 128, ISP 126, local network 122 and communication interface 118.

The received code may be executed by processor 104 as it is received, and/or stored in storage device 110, or other non-volatile storage for later execution. In this manner, computer system 100 may obtain application code in the form of a carrier wave.

FUNCTIONAL OVERVIEW OF APPLICATION SERVER

Figure 2 is a block diagram of a system 200 designed according to an embodiment of the invention. The system 200 includes a plurality of browsers 202, 204 and 206 that communicate with a plurality of listeners 210, 216 and 222 over the Internet 208 according to the HTTP protocol. In response to requests from the browsers, the listeners cause a web application server 280 to invoke software modules, referred to herein as cartridges. In the illustrated embodiment, web application server 280 has initiated the execution of three cartridges 230, 234 and 238.

The web application server 280 is composed of numerous components, including transport adapters 212, 218 and 224, dispatchers 214, 220 and 226, an authentication server 252, a virtual path manager 250, a resource manager 254, a configuration provider 256 and a plurality of cartridge execution engines 228, 232 and 236. The various components of the web application server 280 shall be described hereafter in greater detail.

Significantly, the numerous components of web application server 280 communicate through an inter-machine communication mechanism, such as an Object Request Broker 282. Using an inter-machine communication mechanism, cartridge instances that perform the operations specified in browser requests may execute on different machines than the listeners

that receive the requests and the browsers that issue the requests. Because the cartridge instances are on different machines than the listeners, the listeners are better insulated against faulty cartridge instances, thus enhancing the reliability and security of the system. In addition, the scalability of the system is greatly increased by spreading the processing burden of executing the cartridge instances among many machines, rather than the same machine that is executing the listener. The ability to distribute cartridge instance execution across multiple machines allows numerous types of load balancing techniques to be used in deciding when and where to spawn new cartridge instances.

A typical operation within system 200 generally includes the following stages:

A browser transmits a request over the Internet 208.

A listener receives the request and passes it through a transport adapter to a dispatcher.

The dispatcher communicates with the virtual path manager 250 to determine the appropriate cartridge to handle the request.

At this point the dispatcher does one of two things. If the dispatcher knows about an unused instance for that cartridge, the dispatcher sends the request to that instance. If there are no unused cartridge instances for that cartridge, the dispatcher asks the resource manager 254 to create a new cartridge instance. After the instance starts up successfully, the cartridge notifies the resource manager of its existence. The resource manager 254 then notifies the dispatcher of the new instance. The dispatcher creates a revised request based on the browser request and sends the revised request to the new instance.

The cartridge instance handles the revised request and sends a response to the dispatcher.

The dispatcher passes the response back through the listener to the client.

These stages shall be described in greater detail hereafter.

CARTRIDGES

Cartridges are modules of code for performing specific application or system functions. A cartridge forms the basic unit of distribution in the system 200. According to one embodiment of the invention, cartridges are named using Universal Resource Locators (URLs). Thus, a cartridge name has two parts: the IP address of the server on which the cartridge resides, and the virtual path in the server directory structure of the compiled cartridge code. Because cartridges are named using URLs, the cartridge name space is global

and cartridges may be accessed using the same messaging techniques as are used to access other web resources, such as documents.

According to one embodiment of the invention, each cartridge has a standard interface which provides a common overall structure for all cartridges. The standard interface defines the interface of routines that are invoked by the web application server 280 under particular conditions. According to one embodiment of the invention, the abstract cartridge interface is as follows:

```
interface Cartridge
   {
boolean init();
boolean authenticate(in Principal user_passwd);
boolean exec(in Request req_obj, out Response resp_obj);
boolean shutdown();
}
```

The init() routine is responsible for intializing the cartridge instance. This may include invoking the constructors of several subobjects, preforking threads and acquiring all other required shared resources.

The shutdown() routine is responsible for cleaning up all of the resources and shutting down the cartridge instance. Once the shutdown() routine is invoked on a cartridge instance, it immediately becomes unavailable for servicing subsequent requests.

The authenticate() routine validates whether the client requesting the services of the cartridge is authorized to use those services.

The exec() routine is the generic way to dispatch all service requests to the cartridge.

EXEMPLARY CARTRIDGES

Each cartridge is either configured as a cartridge that performs a well-defined function, or as a programmable cartridge that acts as an interpreter or a routine environment for an application. An example of a programmable cartridge is a PL/SQL runtime, configured to process database queries according to the Oracle-based Programming Language using Structured Query Language (PL/SQL). The PL/SQL runtime executes a browser request having a database query. The PL/SQL runtime processes the request, for example, by accessing a database server in communication with the cartridge instance via a data link.

Another example of a programmable cartridge is a JAVA runtime interpreter. The JAVA runtime interpreter cartridge enables web application developers to write server-side JAVA applications to process browser requests. Similarly, a custom server may be configured as a cartridge in order to provide dynamic operations such as, for example, accessing processes executed by a third party server.

DISPATCHERS

Dispatchers are software modules configured to route the requests received by listeners to the appropriate cartridges. According to one embodiment of the invention, dispatchers are implemented as server-side program extensions (i.e. "plug-ins"). As such, the dispatchers are loaded into and execute within the same address space as the listeners to which they belong. The dispatchers may be linked with the listener code at compile time or dynamically loaded at runtime.

In the illustrated embodiment, dispatchers 214, 220 and 226 are associated with listeners 210, 216 and 222, respectively. Dispatchers 214, 220 and 226 selectively route browser requests received by listeners 210, 216 and 222 to cartridges.

For example, assume that listener 210 receives a browser request over the Internet 208 delivered in the form of a Uniform Resource Locator (URL). The browser request serves as an identifier for a web object, for example an HTML page or an operation to be performed. The listener 210 hands off the browser request to dispatcher 214 without any attempt at interpreting the browser request. Upon receiving the browser request, the dispatcher 214:

(1) communicates with virtual path manager 250 to identify a cartridge selected by the browser request and to determine whether the cartridge requires authentication,

(2) if the cartridge requires authentication, communicates with the authentication server 252 to determine whether the browser is allowed to access the selected cartridge,

(3) if access is authorized, communicates with the resource manager to determine the specific instance of the selected cartridge to which the browser request should be sent, and

(4) creates and dispatches a revised browser request for execution by the specified instance of the cartridge.

The revised browser request repackages information received in the original browser request. The revised browser request may include, for example, a context object that contains data required for the proper operation of the cartridge. The data required for proper operation

of a cartridge may include, for example, a transaction ID that identifies a transaction with which the browser request is associated.

If the cartridge replies to the request, the cartridge sends the reply to the dispatcher and the dispatcher passes the reply up to the listener for transmission to the browser that initiated the request.

CONFIGURATION PROVIDER

According to one embodiment of the invention, cartridges that are to be used with web application server 280 are first registered with web application server 280. During the registration process, information about the cartridges is supplied to the configuration provider 256. Configuration provider 256 stores the information as metadata 258 for later access by the components of the web application server 280.

The metadata 258 may include, for example,

(1) the cartridge name;

(2) the minimum number of required instances;

(3) the maximum number of instances;

(4) the location of the code that implements the cartridge;

(5) the program-dependent function names used by the cartridge execution engine to execute the callback functions (initialization, request handler, shutdown);

(6) a list of machines for running the cartridge;

(7) the idle time for the cartridge (the amount of time instances of the cartridge are allowed to remain idle before they are shut down);

(8) an object identifier; and

(9) data indicating the type of authentication service, if any, to be used with the cartridge.

The object identifier specifies the data that must be supplied by a browser request for requesting performance of an operation by the corresponding cartridge. The object type may be a specific word, a URL, or may include a virtual path such as "/java".

Once the configuration provider 256 has stored the configuration information for a particular cartridge in the metadata 258, that cartridge is automatically registered when web application server 280 is started.

After a cartridge is registered with the web application server 280, the resource manager 254 initiates the minimum instances for the cartridge. Once the minimum number of

instances has been initiated, the web application server 280 is prepared to process browser requests.

THE VIRTUAL PATH MANAGER

As mentioned above, dispatchers communicate with the virtual path manager 250 to determine where to route each revised browser request. Specifically, each browser request typically includes a URL. Upon receiving a browser request, the dispatcher sends the URL in the request to the virtual path manager 250. The virtual path manager 250 responds by sending the dispatcher data that identifies the cartridge, if any, associated with the URL.

In order to supply the required information to dispatchers, virtual path manager 250 consults the metadata 258 that maps URLs to cartridges. In response to receiving a browser request, the virtual path manager 250 uses the mapping data to determine the cartridge, if any, to which the URL contained in the browser requests corresponds.

For example, if the browser request is a URL request beginning with the virtual path "/java", the mapping may indicate that the JAVA interpreter cartridge is configured to handle requests having the virtual path "/java".

According to one embodiment of the invention, the virtual path manager 250 also determines whether the cartridge associated with the URL requires authentication. If the cartridge requires authentication, the virtual path manager 250 indicates in the response that the virtual path manager 250 sends to the dispatcher that authentication is required. If authentication is not required, the dispatcher creates and sends a revised browser request to an instance of the cartridge without invoking the authentication server 252. If authentication is required, the dispatcher sends the revised request to an instance of the cartridge only after the authentication server indicates that the revised request may be submitted to an instance of the cartridge.

THE RESOURCE MANAGER

The resource manager 254 of the web application server 280 manages the execution of each of the cartridges by initiating a predetermined minimum number of instances for the cartridges, load balancing between the instances of each cartridge, and initiating new instances of cartridges as necessary up to a predetermined maximum number of instances of a given cartridge.

For example, assume that the metadata for a particular cartridge (C1) includes the following information:

Name = C1

Minimum Instances = 10

Maximum Instances = 50

Host Machines = M1, M3, M8

Idle time = 30 seconds

Based on this metadata, when cartridge C1 is first registered, resource manager 254 will initiate ten instances of C1. Resource manager 254 will initiate the ten instances on the machines associated with the labels M1, M3 and M8.

Upon receipt of requests from dispatchers to access C1, resource manager 254 determines whether any existing instance of C1 is available for use. If no instance of C1 is available when a request is received, resource manager 254 determines whether the maximum number of instances of C1 are already running. If the maximum number of instances of C1 are not already running, then resource manager 254 initiates a new instance of C1 on one of the possible host machines and transmits a message that identifies the new instance to the dispatcher that issued the request. If the maximum number of instances of C1 are already running, then resource manager 254 sends a message to the dispatcher that issued the request to indicate that the request cannot be handled at this time.

LOAD BALANCING

According to one embodiment of the invention, resource manager 254 applies a set of load balancing rules to determine where to initiate instances of cartridges where there is more than one possible host machine. Thus, in the above example, M1, M2 and M3 are all capable of executing instances of cartridge C1. If M1, M2 and M3 have the same processing capacity, it may be desirable to distribute the instances evenly across the three machines. However, if M1 has ten times the processing power of M2 and M3, it may be desirable to initiate all instances of C1 on M1 up to a certain point, and then to distribute additional instances evenly among M1, M2 and M3.

To assist resource manager 254 in determining how to load balance among possible machines, the metadata stored for each cartridge may include additional details. For example, the metadata may specify a separate minimum and maximum number of instances for each

machine. Resource manager 254 may then distribute new instances among the machines based on which machine has the lowest ratio of actual instances to maximum instances.

The metadata may also specify an order for the machines that can run a cartridge. The machine at the N+1 position in the order is only used to execute instances of the cartridge when the machine at the Nth position in the order is already executing its maximum number of instances.

CARTRIDGE INSTANCE STATUS TRACKING

According to one embodiment of the invention, the resource manager 254 maintains state information to keep track of cartridge instances that have been created. The state information includes data that identifies the instance, identifies the machine executing the instance, and identifies the listener to which the instance has been assigned.

Figure 5 illustrates a table 500 that may be maintained by resource manager 254 to store this state information. Table 500 includes an instance column 502, a cartridge column 504, a listener column 506 and a machine column 508. Each row of table 500 corresponds to a distinct cartridge instance. Within the row for a given cartridge instance, cartridge column 504 identifies the cartridge associated with the cartridge instance and instance column 502 indicates the instance number of the cartridge instance. For example, row 510 corresponds to an instance of cartridge C1. Therefore, cartridge column 504 of row 510 indicates cartridge C1. Instance column 502 of row 510 indicates that the cartridge instance associated with row 510 is instance 1 of cartridge C1.

Listener column 506 indicates the listener to which the cartridge instance associated with a row has been assigned. Machine column 508 indicates the machine on which the cartridge instance associated with a row is executing. For example, the cartridge instance associated with row 510 has been assigned to listener 210 and is executing on machine M1.

Similar to resource manager 254, each dispatcher maintains state information for the cartridge instances that have been assigned to the listener to which the dispatcher is attached. Such state information may be maintained, for example, in a table 400 as shown in Figure 4. Similar to table 500, table 400 includes an instance column 402 and a cartridge column 404 that respectively hold instance numbers and cartridge identifiers. However, while table 500 includes one entry for every cartridge instance assigned by resource manager 254, table 400 only includes entries for cartridge instances that have been assigned to a particular listener.

For example, table 400 includes entries for only those cartridge instances listed in table 500 that have been assigned to listener 210.

In addition to instance column 402 and cartridge column 404, table 400 includes a status column 406. For each row, the status column 406 holds a value that indicates the status of the instance associated with the row. For example, the status column 406 of row 408 indicates that instance 1 of cartridge C1 is currently busy. In the illustrated embodiment, the status column 406 holds a flag that indicates that a cartridge instance is either BUSY or FREE. The significance of the cartridge status shall now be describe with reference to the operation of resource manager 254 and dispatchers 214 and 220.

INTERACTION BETWEEN DISPATCHERS AND THE RESOURCE MANAGER

As explained above, dispatchers communicate with resource manager 254 when they need to send a revised browser request to a particular cartridge. According to one embodiment of the invention, dispatchers first determine whether an instance of the appropriate cartridge (1) has already been assigned to it and (2) is available to process the new revised browser request. If an appropriate cartridge instance has already been assigned to the dispatcher and is currently available to process the new revised browser request, then the dispatcher forwards the revised browser request to the cartridge instance without further communication with resource manager 254.

For example, assume that listener 210 receives a browser request that, according to virtual path manager 250, must be processed by cartridge C1. Assume also that table 400 reflects the current list and status of cartridge instances that have been assigned to listener 210. Upon receiving the browser request from listener 210, dispatcher 214 inspects table 400 to locate a FREE instance of cartridge C1. In the illustrated table 400, row 410 indicates that instance 3 of cartridge C1 is currently FREE. Consequently, dispatcher 214 forwards a revised browser request directly to instance 3 of cartridge C1 without further communication with resource manager 254. In response to sending the revised browser request, dispatcher 214 changes the status value in status column 406 of row 410 to BUSY.

If a listener has not already been assigned an appropriate cartridge instance that is currently available, then the dispatcher associated with the cartridge requests a cartridge instance from the resource manager 254. If the resource manager 254 determines that an instance of the required cartridge is not available and the number of existing instances of the

required cartridge is below the maximum, then the resource manager 254 initiates a new cartridge. Upon initiating a new cartridge, the resource manager 254 inserts an entry for the new cartridge instance in table 500.

Assume, for example, that listener 210 receives a browser request that must be processed by cartridge C3. Assume also that instance 3 of cartridge C3 has not yet been initiated. Under these conditions, dispatcher 214 sends to resource manager 254 a request for a handle to an instance of cartridge C3. In response to this request, resource manager 254 initiates instance 3 of cartridge C3 on machine M3. In addition, resource manager 254 inserts into table 500 the entry found at row 512.

After inserting row 512 for instance 3 of cartridge C3 in table 500, resource manager 254 sends back to the dispatcher 214 a handle to the newly created instance. In response to receiving this handle, dispatcher 214 inserts an entry (row 412) for the new instance in its status table 400. The dispatcher 214 then transmits a revised browser request to instance 3 of cartridge C3.

RELEASING CARTRIDGE INSTANCES

According to one embodiment of the invention, listeners do not automatically release ownership of cartridge instances when the cartridge instances finish responding to outstanding browser requests. For example, assume that instance 3 of cartridge C3 receives a revised browser request, processes the revised browser request, and sends a response back to dispatcher 214. Dispatcher 214 passes the response to listener 210 to be sent back to the browser that issued the browser request.

At this point, listener 210 no longer requires ownership of instance 3 of cartridge C3. However, rather than transferring ownership of instance 3 of cartridge C3 back to resource manager 254, dispatcher 214 merely changes the status column 406 of row 412 from BUSY to FREE.

Changing the value in status column 406 of row 412 to FREE indicates that instance 3 of cartridge C3 is no longer working on a request, and is therefore ready to handle subsequent requests. However, because table 400, which indicates that instance 3 of cartridge C3 is available, is maintained locally by dispatcher 214, instance 3 of cartridge C3 is only available for subsequent browser requests arriving at listener 210. Row 512 of table 500 maintained by resource manager 254 continues to indicate that instance 3 of cartridge C3 is owned by listener 210.

Because listeners do not automatically release cartridge instances every time a request is serviced, overhead associated with communication between the resource manager 254 and the various dispatchers is significantly reduced. For example, assume that a listener 210 receives ten successive requests that must be communicated to cartridge C3. Rather than communicating with resource manager 254 for each of the ten requests, dispatcher 214 may communicate with resource manager 254 in response to the first request. The subsequent nine requests can be handled by dispatcher 214 without communicating with resource manager 254 because the dispatcher 214 uses the same instance of C3 that processes the first request to process the nine subsequent requests.

While not automatically releasing listener ownership of cartridge instances when each request is serviced can increase the efficiency of web application server 280, listeners cannot maintain ownership of cartridge instances indefinitely. For example, instances that have not been used for long periods of time should be passed back to the resource manager 254 so they can be de-allocated to free up resources. In addition, it is not efficient for one listener to maintain ownership of the instance of a cartridge that it has not used for a relatively long time when other listeners require instances of that cartridge.

Consequently, resource manager 254 communicates to each listener a maximum idle time for each cartridge instance passed to the listener. The maximum idle time indicates the maximum amount of time a cartridge instance can go unused before the listener must release ownership of the cartridge instance. For example, assume that the resource manager 254 indicates to listener 210 that the maximum amount of idle time for instance 3 of cartridge C3 is 10 minutes. Based on this information, listener 210 may continue to use instance 3 of cartridge C3 to process browser requests for cartridge C3 as long as instance 3 of cartridge C3 does not remain idle or FREE for more than 10 minutes.

If instance 3 of cartridge C3 is idle for more than 10 minutes, dispatcher 214 removes row 412 from table 400 and sends a message to resource manager 254 that listener 210 is releasing ownership of instance 3 of cartridge C3. In response to this message, resource manager 254 updates row 512 to indicate that instance 3 of cartridge C3 is not owned by any listener and may thus be reassigned to another listener or terminated.

In an alternative embodiment, dispatchers do not automatically release cartridge instances when the idle time for the cartridge instance has expired. Instead, the dispatcher sends a message to resource manager 254 offering to release the expired instance. Resource

manager 254 may respond to the offer by requesting that the listener release the cartridge instance, or by allowing the listener to retain ownership of the expired cartridge instance.

According to one embodiment of the invention, resource manager 254 maintains a queue of the requests that cannot be immediately serviced. When it becomes possible to service a queued request, the request is removed from the queue and processed.

For example, assume that listener 222 receives a browser request that must be processed by cartridge C1, and that listener 222 has not been assigned any instances of cartridge C1. Dispatcher 226 sends a request for an instance of C1 to resource manager 254. Assume further that a maximum of 50 instances of C1 are allowed, and that 50 instances of C1 have been assigned to listener 210. Under these conditions, resource manager 254 cannot service the request from listener 222. Therefore, resource manager 254 puts the request on a queue. When listener 210 releases an instance of C1, resource manager 254 communicates to listener 222 that an instance of C1 is available.

Under certain conditions, resource manager 254 may preemptively cause a listener to release a cartridge instance. For example, resource manager 254 may detect a system overload situation and respond by terminating a set of cartridge instances, either before or after informing the listeners that currently have been assigned the cartridge instances that the cartridge instances are going to be terminated.

Resource manager 254 may also preemptively cause listeners to release cartridge instances to implement fairness policies between listeners. For example, resource manager 254 may cause a listener that holds the most instances of a given cartridge to release an instance of the cartridge when another listener has waited more than a predetermined threshold of amount of time for an instance of the cartridge. For example, if listener 210 has been assigned 50 instances of cartridge C1 and C1 has a maximum of 50 instances, then resource manager 254 may cause listener 210 to release an instance of C1 ten seconds after receiving a request for an instance of C1 from another listener.

CARTRIDGE EXECUTION ENGINES

According to one embodiment of the invention, each cartridge instance is composed of a cartridge execution engine and a cartridge. A cartridge execution engine is a code module that insulates cartridges from the complexities of the web application server 280 and the inter-module communication mechanism. A cartridge is made available to a cartridge execution engine by storing in a function table pointers to the cartridge functions. According to one

embodiment, all cartridges provide the functions specified in the exemplary cartridge interface described above. By having all cartridges support the same interface, a single standard cartridge execution engine can be used with all cartridges.

According to one embodiment of the invention, cartridges are implemented as shared libraries, and cartridge execution engines are executable programs that invoke the routines in the shared libraries using the standard cartridge interface. The cartridge execution engine provides the interface between cartridges and the dispatcher, directs cartridge flow of control, and provides services for cartridges to use.

When the resource manager 254 requires the creation of a new cartridge instance, the resource manager 254 causes a cartridge execution engine to be instantiated. In turn, the instance of the cartridge execution engine thus created causes the appropriate cartridge to be instantiated. The resource manager 254 can cause the cartridge execution engine to be instantiated, for example, by invoking a "cartridge execution engine factory" that resides on the machine on which the cartridge is to be executed. The instance of the cartridge execution engine can cause the cartridge to be instantiated, for example, by making a call to one of the routines in the shared library that constitutes the cartridge.

As shown in Figure 2, the web application server 280 includes cartridge execution engines 228, 232 and 236 for each of the cartridges 230, 234 and 238. The cartridge execution engines control execution of the instances of the corresponding cartridges by making calls into the cartridges through the standard cartridge interface. By establishing basic callback functions between the cartridge execution engine and a cartridge, any cartridge can be integrated into the web application server 280 by configuring the cartridge to respond to the callback functions, and then registering the cartridge in the configuration provider 256, as described below.

Thus, if the dispatcher 214 determines that the PL/SQL runtime cartridge is the appropriate cartridge to process a request, the dispatcher 214 dispatches the request to a cartridge instance that includes a cartridge execution engine associated with the PL/SQL runtime cartridge. If a new instance needs to be initiated, the resource manager 254 creates a new instance of the PL/SQL runtime cartridge in a separate address space and dispatches the request to the cartridge execution engine 228 of the new instance. The address space used to execute the instance of the program may be within memory of the computer system upon which one or more of the components of web application server 280 is executing, or on another computer system.

In response to a message from a dispatcher, the cartridge execution engine issues a request handler callback function to the cartridge, causing the cartridge to process the request. The cartridge executing the request returns the result to the cartridge execution engine, which forwards the result to the dispatcher. In the event that the web application server 280 detects a fault in the operation, the cartridge execution engine issues a shutdown function of the cartridge.

Hence, the cartridge execution engine provides an application programming interface to the web application server 280 that specifies predetermined operations to be performed. Use of the standard cartridge interface enables programmers of the cartridges to configure each cartridge for high-level integration into the web application server 280 independent of the protocols used by the particular web listener with which the cartridge will be used.

TRANSPORT ADAPTERS

Listeners enable the use of server-side plug-ins by providing a programming interface and protocol for use by such plug-ins. Unfortunately, the programming interfaces and protocols provided by listeners vary from listener to listener. For example, Netscape Server Application Programming Interface (NSAPI), Internet Server Application Programming Interface (ISAPI) and Application Development Interface (ADI) are three examples of distinct programming interfaces currently provided by listeners.

Transport adapters insulate dispatchers from the proprietary protocols and interfaces used by web listeners. Specifically, each transport adapter is configured to recognize the protocols of different listeners, and to convert the browser requests received from the listeners into converted browser requests having a standard dispatcher protocol that is independent from the protocol of the listener. Similarly, transport adapters convert the replies from the dispatcher to the transport protocol of the listeners.

Hence, the transport adapter enables the web application server 280 to be used with listeners from different vendors. Moreover, transport adapters may be configured to accommodate different server architectures and operating systems.

OPERATION OF THE WEB APPLICATION SERVER

Figures 3A and 3B are a flow diagram illustrating a method of responding to a browser request according to an embodiment of the present invention. The browser request is received

in step 350 by a listener. For the purposes of explanation, it shall be assumed that the browser request was issued by browser 202 and received by listener 210.

Upon receiving the browser request, the listener 210 forwards the request to the web application server 280 in step 352. Specifically, listener 210 passes the request to the transport adapter 212 using the proprietary programming interface of the listener 210. The transport adapter 212 converts the request as necessary to pass the request to dispatcher 214 using a standard dispatcher programming interface.

Dispatcher 214 identifies the request object type that corresponds to the browser request in step 354 based on the virtual path specified in the browser request by communicating with the virtual path manager 250. The dispatcher 214 determines in step 356 if the request object type corresponds to an identifiable cartridge. If the request object type does not correspond to an identifiable cartridge, the request is returned to the listener 210 in step 358 (see Figure 3B). If in step 358 the listener 210 recognizes the request as a request for a static HTML page, the listener accesses the static HTML page, and sends the HTML page to the browser 202 in step 360. If the browser request is not recognized by the listener 210, the reply is sent to the browser 202 in step 360 indicating that the request was unrecognizable.

If in step 356 the dispatcher 214 determines that (1) the request must be sent to a cartridge and (2) listener 210 has not been assigned any instances of that cartridge that are currently FREE, then the dispatcher 214 communicates with the resource manager 254 to be assigned an instance of the cartridge 230 to which the browser request can be sent. In step 362, shown in Figure 3B, the resource manager 254 determines whether an instance of the identified cartridge is available (unowned) among the existing number of instances. For the purposes of explanation, it shall be assumed that the request is associated with cartridge 230, and that cartridge 230 is a PL/SQL runtime cartridge.

If in step 362 the resource manager identifies an available instance, for example instance 260 of the PL/SQL runtime 230, the resource manager 254 informs the dispatcher 214 that the request should be sent to instance 260. The dispatcher 214 then creates and sends a revised browser request to the cartridge execution engine 228 of the instance 260 in step 368 to cause the available instance 260 to process the request, as described below.

However, if in step 362 no instance of the cartridge 230 is available, the resource manager 254 determines in step 364 if the existing number of instances exceeds a maximum prescribed number. If the existing number of instances exceeds the maximum prescribed number in step 364, the resource manager 254 indicates to the dispatcher 214 that the request

cannot be processed at this time. In response, the dispatcher 214 returns the request to the listener 210 in step 358, after which the web listener 210 sends a reply to the browser 202 over the network in step 360 indicating the request was not processed.

Alternatively, when a cartridge instance is not currently available to handle a request, listener 210 may place the request on a waiting list for that cartridge instance. When a cartridge instance becomes available, the revised browser request is removed from the waiting list and forwarded to the cartridge instance. If the revised browser request remains on the waiting list for more than a predetermined amount of time, listener 210 may remove the request from the waiting list and send a message to the browser 202 to indicate that the request could not be processed.

If in step 364 the existing number of instances does not exceed the maximum prescribed number, the resource manager 254 initiates a new instance of the identified program and informs the dispatcher 214 that a revised browser request based on the browser request should be sent to the new instance. The dispatcher 214 then dispatches a revised browser request to the cartridge execution engine of the new instance.

For example, assume that the resource manager 254 initiated instance 260 in response to the browser request. During the initialization, the stored sequences of instructions for the PL/SQL runtime are accessed to create a new instance 260 of the cartridge 230 in an address space that is separate from the address space in which dispatcher 214 is executing. According to one embodiment, initialization is performed by loading the cartridge execution engine 228 and having the cartridge execution engine call the initialization routine in cartridge 230.

Once the new instance 260 is running, the dispatcher 214 dispatches the request to the cartridge execution engine 228 associated with the new instance 260 in step 368. The cartridge execution engine 228 sends a callback message to the new instance 260 requesting execution of the request. In the callback message, the cartridge execution engine 228 passes any parameters necessary for the instance 260 to process the request. Such parameters may include, for example, passwords, database search keys, or any other argument for a dynamic operation executed by the instance 260.

The instance 260 then executes the request. During the execution of the request by the instance in step 368, the dispatcher 214 monitors the instance to determine the occurrence of a fault in step 370. If in step 370 the dispatcher 214 detects a fault, the dispatcher 214 calls the corresponding cartridge execution engine 228 in step 372 to abort the instance 260 having the fault. The corresponding cartridge execution engine 228 in turn issues a shut down command

across the API to the faulty instance. The instance, responding to the shut down command by the cartridge execution engine 228, will shut down without affecting any other process in any other address space.

If in step 370 no fault is detected, the dispatcher 214 receives a reply from the instance 260 upon completion of execution in step 374. The dispatcher 214 in step 376 forwards the reply to the listener 210, which responds to the browser with the reply from the executed instance 260. After executing the instance 260, the dispatcher 214 in step 378 maintains the instance in the memory, as shown in step 378 to enable execution of a subsequent request.

DISTRIBUTED ARCHITECTURE OF WEB SERVER

Significantly, the various components of the web application server 280 communicate with each other using a communication mechanism that does not require the components to be executing in the same address space or even on the same machine. In the illustrated embodiment, the components of the web application server 280 are configured to communicate through an Object Request Broker (ORB) 282. Object Request Brokers are described in detail in "Common Object Request Broker: Architecture and Specification (CORBA)". This and other documents relating to CORBA can be found on the World Wide Web at http://www.omg.org.

While the embodiments of the present invention shall be described with reference to communications through a CORBA-compliant ORB, other cross-platform communication mechanisms may be used. For example, the components of web application server 280 may alternatively communicate with each other using Remote Procedure Calls (RPC), a UNIX pipe, Microsoft COM.

Because the various components of the web application server 280 communicate with each other using a machine independent communication mechanism, there are no inherent restrictions with respect to where the components are located with respect to each other. For example, listeners 210, 216 and 222 may be executing on the same machine, or on three completely different machines, each with a different operating system. Similarly, the authentication server 252, virtual path manager 250, resource manager 254 and configuration provider 256 may be executing on the same machine or on four different machines. Further, those four different machines may not have any overlap with the three machines executing listeners 210, 216 and 222.

Cartridge execution engines 228, 232 and 236 incorporate all of the necessary logic to communicate with the other components of the web application server 280 through the object request broker 282. Consequently, the location of the cartridge instances themselves is not inherently restricted by the communication mechanism. Thus, instance 260 may be executing in a completely different machine and operating system than dispatchers from which it receives requests. Likewise, instance 260 may be on a different machine and operating system than the resource manager 254 or any of the other components of the web application server 280, including instances of other cartridges that are being managed by the same web application server 280.

Significantly, the location-independence enjoyed by cartridges used by web application server 280 is achieved through the cartridge execution engine communication logic, not through any custom programming in the cartridges themselves. Consequently, the cartridges do not need to be specially designed for execution in a distributed application server environment. Cartridge designers are thus insulated from the complexities of a distributed system, and can concentrate their efforts on the logic associated with the tasks for which the cartridges were created.

As described above, a web application server 280 that manages multiple instances of different cartridges to process a variety of user requests is provided. Each cartridge instance executes in a separate memory space from the listener, thus avoiding the security problems associated with server-side plug-ins. Further, the cartridge instances used to process the browser requests received by a listener may execute on entirely different machines than the listener. Thus, the scalability of the system is increased, while the burden on any particular machine is reduced.

In addition, web application server 280 also controls the number of instances for each given cartridge. Hence, server-side resources are controlled to ensure that a large number of requests do not overwhelm any machine by an uncontrollable generation of instances.

Further, execution throughput also is improved by maintaining a minimum number of instances ready for execution. Additional instances may be initiated and maintained in memory for executing subsequent requests, as opposed to terminating an instance after a single execution and then reloading the cartridge into memory in order to recreate an instance for execution of a subsequent request.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and

changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A method for executing an operation, the method comprising the steps of:
executing a dispatcher on a first machine;
executing a resource manager on a second machine;
sending a first message from the dispatcher to the resource manager, where the first message identifies a particular cartridge that is able to perform said operation;
sending a second message from the resource manager to the dispatcher, where the second message identifies a particular instance of the particular cartridge, where the particular instance is executing on a third machine; and
sending a third message from the dispatcher to the particular instance to cause the particular instance to execute the operation; and
wherein at least two of the first machine, the second machine and the third machine are separate machines.

2. The method of Claim 1 further comprising the steps of:
a browser sending a browser request to a web listener;
the web listener passing the browser request to the dispatcher;
wherein the dispatcher sends the first message in response to receiving the browser request from the web listener.

3. The method of Claim 1 wherein:
the first machine and the second machine are separate machines;
the step of sending a first message from the dispatcher to the resource manager is performed by sending the first message from the dispatcher to the resource manager through an object request broker; and
the step of sending a second message from the resource manager to the dispatcher is performed by sending the second message from the resource manager to the dispatcher through the object request broker.

4. The method of Claim 1 wherein:
the second machine and the third machine are separate machines; and

the method includes the step of the resource manager causing the particular instance to begin execution on the third machine in response to the resource manager receiving the first message.

5. The method of Claim 1 wherein the resource manager determines which particular instance should perform the operation by performing the following steps in response to receiving the first message:

determining whether any instance of the particular cartridge is currently executing and unassigned;

if any instance of the particular cartridge is currently executing and unassigned, then identifying in said second message an instance that is currently executing and unassigned;

if no instance of the particular cartridge identified in the first message is currently executing and unassigned, then initiating a new instance of the particular cartridge and identifying said new instance in said second message.

6. The method of Claim 5 wherein, prior to initiating the new instance, the resource manager performs the steps of:

inspecting metadata to determine a set of machines, including said third machine, that are able to execute said particular cartridge; and

selecting said third machine to execute said new instance of said particular cartridge.

7. The method of Claim 1 further comprising the steps of:

causing said dispatcher to maintain a list of instances that have been assigned to said dispatcher; and

in response to receiving said second message, causing said dispatcher to update said list of instances to include an entry for said particular instance.

8. The method of Claim 7 wherein:

the list of instances includes data indicating whether the instances in said list are currently busy;

in response to sending said second message, the dispatcher indicates in said entry for said particular instance that said particular instance is busy;

the method further includes the step of the dispatcher receiving a fourth message from said particular instance in response to said third message; and

in response to receiving said fourth message, the dispatcher updates said entry to indicate that said particular instance in not currently busy.

9. The method of Claim 8 further comprising the steps of:

the dispatcher receiving a browser request that specifies a second operation to be performed by a second cartridge;

the dispatcher inspecting said list of instances to determine whether the list contains an entry for an instance of said second cartridge that is not currently busy;

if the list contains an entry for an instance of said second cartridge that is not currently busy, performing the steps of

the dispatcher sending a fifth message to said particular instance in response to said browser request to cause said particular instance to perform said second operation; and

updating said entry to indicate that said particular instance is currently busy;

if the list does not contain an entry for an instance of said second cartridge that is not currently busy, sending a sixth message to said resource manager to cause said resource manager to indicate an instance of said second cartridge to perform said second operation.

10. The method of Claim 8 further comprising the steps of:

determining when said particular instance has been idle for more than a predetermined length of time; and

when said particular instance has been idle for more than a predetermine length of time, said dispatcher releasing said particular instance to said resource manager.

11. A computer-readable medium carrying one or more sequences of instructions for executing an operation, wherein execution of the one or more sequences of instructions by one or more processors causes the one or more processors to perform the steps of:

executing a dispatcher on a first machine;

executing a resource manager on a second machine;

sending a first message from the dispatcher to the resource manager, where the first message identifies a particular cartridge that is able to perform said operation;

sending a second message from the resource manager to the dispatcher, where the second message identifies a particular instance of the particular cartridge, where the particular instance is executing on a third machine; and

sending a third message from the dispatcher to the particular instance to cause the particular instance to execute the operation; and

wherein at least two of the first machine, the second machine and the third machine are separate machines.

12. The computer-readable medium of Claim 11 further comprising instructions for performing the steps of:

a web listener passing a browser request received from a browser to the dispatcher;

wherein the dispatcher sends the first message in response to receiving the browser request from the web listener.

13. The computer-readable medium of Claim 11 wherein:

the first machine and the second machine are separate machines;

the step of sending a first message from the dispatcher to the resource manager is performed by sending the first message from the dispatcher to the resource manager through an object request broker; and

the step of sending a second message from the resource manager to the dispatcher is performed by sending the second message from the resource manager to the dispatcher through the object request broker.

14. The computer-readable medium of Claim 11 wherein:

the second machine and the third machine are separate machines; and

the computer-readable medium includes the step of the resource manager causing the particular instance to begin execution on the third machine in response to the resource manager receiving the first message.

15. The computer-readable medium of Claim 11 wherein the resource manager determines which particular instance should perform the operation by performing the following steps in response to receiving the first message:

determining whether any instance of the particular cartridge is currently executing and unassigned;

if any instance of the particular cartridge is currently executing and unassigned, then identifying in said second message an instance that is currently executing and unassigned;

if no instance of the particular cartridge identified in the first message is currently executing and unassigned, then initiating a new instance of the particular cartridge and identifying said new instance in said second message.

16. The computer-readable medium of Claim 15 wherein, prior to initiating the new instance, the resource manager performs the steps of:

inspecting metadata to determine a set of machines, including said third machine, that are able to execute said particular cartridge; and

selecting said third machine to execute said new instance of said particular cartridge.

17. The computer-readable medium of Claim 11 further comprising instructions for performing the steps of:

causing said dispatcher to maintain a list of instances that have been assigned to said dispatcher; and

in response to receiving said second message, causing said dispatcher to update said list of instances to include an entry for said particular instance.

18. The computer-readable medium of Claim 17 wherein:

the list of instances includes data indicating whether the instances in said list are currently busy;

in response to sending said second message, the dispatcher indicates in said entry for said particular instance that said particular instance is busy;

the computer-readable medium further includes instructions to perform the step of the dispatcher receiving a fourth message from said particular instance in response to said third message; and

in response to receiving said fourth message, the dispatcher updates said entry to indicate that said particular instance in not currently busy.

19. The computer-readable medium of Claim 18 further comprising instructions for performing the steps of:

the dispatcher receiving a browser request that specifies a second operation to be performed by a second cartridge;

the dispatcher inspecting said list of instances to determine whether the list contains an entry for an instance of said second cartridge that is not currently busy;

if the list contains an entry for an instance of said second cartridge that is not currently busy, performing the steps of

the dispatcher sending a fifth message to said particular instance in response to said browser request to cause said particular instance to perform said second operation; and

updating said entry to indicate that said particular instance is currently busy;

if the list does not contain an entry for an instance of said second cartridge that is not currently busy, sending a sixth message to said resource manager to cause said resource manager to indicate an instance of said second cartridge to perform said second operation.

20. The computer-readable medium of Claim 18 further comprising instructions for performing the steps of:

determining when said particular instance has been idle for more than a predetermined length of time; and

when said particular instance has been idle for more than a predetermine length of time, said dispatcher releasing said particular instance to said resource manager.

21. A system for performing operations associated with browser requests, the system comprising:

a plurality of dispatchers coupled to a plurality of web listeners, wherein each dispatcher of said plurality of dispatchers receives from a corresponding web listener of said plurality of web listeners browser requests received by said corresponding web listener;

a resource manager coupled to said plurality of dispatchers through an inter-machine communication mechanism that allows said resource manager to communicate with said plurality of dispatchers without regard to the machines on which said plurality of dispatchers reside, said resource manager being configured to assign to each dispatcher of said plurality of dispatchers an instance of a cartridge of said plurality of cartridges in response to receiving a request for an instance from said dispatcher;

said plurality of dispatchers being configured to send messages through said inter-machine communication mechanism to the instances that are assigned by said resource manager to said dispatchers, said inter-machine communication mechanism allowing said plurality of dispatchers to communicate with said instances without regard to the machines on which said instances reside, said messages causing said instances to perform the operations associated with said browser requests.

22. The system of Claim 21 further comprising a virtual path manager coupled to said plurality of dispatchers through said inter-machine communication mechanism, said virtual path manager indicating to said dispatchers which of a plurality of cartridges is associated with said browser requests.

23. The system of Claim 21 wherein said inter-machine communication mechanism is an object request broker.

24. The system of Claim 21 wherein the resource manager is further configured to initiate new instances of cartridges in response to messages from said dispatchers.

25. The system of Claim 24 wherein, for each cartridge, said resource manager is configured to maintain a number of executing instances between a predetermined minimum and a predetermined maximum.

26. The system of Claim 21 wherein each dispatcher of said plurality of dispatchers is configured to maintain a list of instances that have been assigned to said dispatcher by said resource manager.

27. The system of Claim 21 wherein, if a dispatcher of said plurality of dispatchers receives a browser request associated with a particular cartridge, and the list of instances maintained by that dispatcher indicates that an instance of that particular cartridge is currently available, then the dispatcher sends a message to the instance without requesting an instance for the browser request from the resource manager.